

Perl & BioPerl Basics

Perl refresher

- Data types
 - Scalars: Strings & numbers
 - Arrays: lists
 - Hashes: Associative arrays, Hashtables, Arrays-where-the-indexes-can-be-non-numbers

Simple Perl

```
my $dna = "GGTGACAGTACG";
my $length = length($dna);
my $reverse = reverse $dna; # reverse
$reverse =~ tr/GCAT/CGTA/; # complement
print "DNA is $length bases long\n";
print "reverse complement of $dna = $reverse\n";

if( $length % 3 != 0 ) {
    print "length is not multiple of 3\n";
} else {
    printf "%s has %d codons\n", $dna, $length / 3;
}
```

Rules to keep life easy

- `use strict;`
`use warnings;`
- `my` to declare variables
- Start simple; Do it in pieces

Strings

- Quotes
 - single quotes - `'literally $3.00'`
 - double quotes - `"value=$var\n"`
 - back quotes - ``execute in shell``
 - formatting -
`$str = sprintf("%s %d", 'formatting', 9);`

String Functions

- upcase, lowercase - `$A = uc($a), $b = lc($B)`
- equality - `if($a eq $b) {...}`
- length - `$len = length($str)`
- substring - `$substr = substr($str,3,5)`
- index of a substring - `$ind = index($str,"b")`
- concatenate - `$sum = $part1 . $part2`
- reverse - `$revstr = reverse($str);`

Easy String Things In Perl

- Append to a string
 - `$str .= "stuff to add at the end";`
- Remove the end if it is whitespace
 - `chomp($str);`
- Type in a bunch of text
 - `$str = qq{ This line and the next line
will all be part of the string};`

Associative Arrays or Hashes

- Key => value pairs
- Like arrays, except can use anything to index, rather than just integers
- No implicit order
- Each key has one value, special tricks to store more than one value per key

Manipulating Hashes

- Add a key,value pair
`$hash{'dog'} = 'beagle';`
- Initialize at once
`%hash = ('dog' => 'beagle',
 'cat' => 'calico');`
- Remove a value
`delete $hash{'cat'};`
- Test if a key has been set
`if(exists $hash{'dog'}) {}`

Hash table functions

- Get the keys

```
@keys = keys %hash;  
for my $key ( @keys ) {  
    print "$key => $hash{$key}\n";  
}
```

- Get the values

```
@values = values %hash;
```

- Loop through both

```
while( ($key,$value) = each %hash) {  
    print "$key => $value\n";  
}
```

Input/Output

- Input

```
open(IN, $filename) || die "cannot open $filename: $!";  
  
while(<IN>) {  
    print "line was $_";  
}
```

- print to output (write) to a data stream. By default it goes to STDOUT.

```
open(OUT, ">$filename") || die "cannot open $filename: $!";  
print OUT "hello\n";  
printf OUT "%d %s\n", 10, "ACT1";  
print OUT join("\t", qw(TAG TGA TAA)), "\n";  
open($fh, ">$newfilename") || die $!;  
print $fh "Filehandle is a variable\n";
```

Input/Output Streams

- STDOUT, STDERR output streams
- STDIN input stream
- `print "hello world\n"` uses STDOUT
- `print STDERR "hello world\n"`
- `open(OUT, ">filename") || die("$filename: $!");`

Filehandles OUTPUT

- `open(FH, ">filename") || die($!);`
- `open($zippedout, "| gzip -c > outfile.gz");`
`print $zippedout "a message\n";`
- `close(FH);`

Filehandles Input

- Diamond operator <>
- ```
open(IN, $filename) || die("$filename:
$!");
while(<IN>) {
 print "line is $_"
}
```
- ```
open($in, "grep 'gene' $datafile |");
while(<$in>) {}
```
- ```
close(IN);
```

# Embedding data in a script

```
while(<DATA>) {
 @line = split;
 next if ($line[0] eq 'SYMBOL');
 print $line[0], " range:",
 $line[2] - $line[3], "\n";
}
```

```
__DATA__
SYMBOL PRICE HIGH LOW
CNN 20.35 25.35 19.75
MSFT 31.27 108.61 29.13
AMZN 37.10 95.21 15.91
```

# Using Modules

- To use a module in your code you need:  
`use List::Util;`
- To use a module which has EXPORTed functions  
`use List::Util qw(sum shuffle);`  
`my @list = (20,32,16,4,28,9,38,36,22);`  
`my @shuffle = shuffle(@list);`  
`my $sum = sum(@list);`

# Object-Oriented Perl

- Objects which encapsulate data and functions
- Modules can encode Classes which define a design plan for an object - architect's *plans*.
- An object is an Instance of a Class - an actual building, and there can be many of them build from the *plans*.

# Objected Oriented Perl

```
use Bio::Seq;
my $str = 'ATGCAATG';
my $id = 'Example1';
my $seq = Bio::Seq->new(-seq => $str,
 -id => $id);

print $seq->display_id, "\n";
print $seq->length, "\n";
print $seq->seq(), "\n";
```

# BioPerl

- Goals of Project
- Simple Example
- Overview of domains addressed
- In depth Example - parsing BLAST

# What is Bioperl?

- A collection of Perl “modules”
- An open-source project to provide reusable Perl code for interfacing with biological data
- Bioinformatics Toolkit - NOT an Application suite
- <http://bioperl.org>
- Stajich et al. (2002) *Genome Res*

# Aims

- Consolidate I-off scripts into something...
  - Reusable
  - Extensible and Modular
  - Fast (enough)
  - Usable for several different skill levels

# Where is it Used?

- Small labs
  - Individual labs collecting benchtop data
  - Students learning informatics programming
- Mid-Large scale Informatics Groups
  - Sequencing centers
  - Academic labs, pharma
- Enterprise
  - Ensembl
  - Generic Model Organism Database (GMOD)

# Quick Example

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;
my $in = new Bio::SeqIO(-format => "genbank",
 -file => "AB077698.gbk");
while(my $seq = $in->next_seq) {
 print "Sequence length is ", $seq->length(),"\n";
 my $sequence = $seq->seq();
 print "1st ATG is at ", index($sequence,ATG)+1,
 "\nfeatures are: \n";
 foreach my $f ($seq->top_SeqFeatures) {
 printf("%s %s(%s..%s)\n", $f->primary_tag,
 $f->strand < 0 ? 'complement' : "",
 $f->start, $f->end);
 }
}
```

LOCUS AB077698 2701 bp mRNA linear PRI 01-MAR-2002  
 DEFINITION Homo sapiens mRNA for hCHCR-G, complete cds.  
 ACCESSION AB077698  
 VERSION AB077698.1 GI:19032344  
 KEYWORDS .  
 SOURCE Homo sapiens (human)  
 ORGANISM Homo sapiens  
 Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;  
 Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.  
 REFERENCE 1  
 AUTHORS Squillace,R.M., Chenault,D.M. and Wang,E.H.  
 TITLE Inhibition of myogenesis by the novel Muscblind-related protein  
 CHCR  
 JOURNAL Unpublished  
 REFERENCE 2  
 AUTHORS Squillace,R.M. and Wang,E.H.  
 TITLE Genomic structure, chromosomal localization, and splicing variation  
 of the human CHCR gene, cloning and characterization of mouse CHCR  
 JOURNAL Unpublished  
 REFERENCE 3 (bases 1 to 2701)  
 AUTHORS Squillace,R.M., Chenault,D.M. and Wang,E.H.  
 TITLE Direct Submission  
 JOURNAL Submitted (10-JAN-2002) Edith H. Wang, University of Washington,  
 Pharmacology; 1959 NE Pacific Ave., Box 357280, Seattle, Washington  
 98195, USA (E-mail:ehwang@u.washington.edu, Tel:206-616-5376,  
 Fax:206-685-3822)  
 FEATURES Location/Qualifiers  
 source 1..2701  
 /organism="Homo sapiens"  
 /db\_xref="taxon:9606"  
 /chromosome="X"  
 /map="Xq26.1"  
 gene 1..2701  
 /gene="CHCR"  
 5'UTR <1..79  
 /gene="CHCR"  
 80..1144  
 CDS /gene="CHCR"  
 /note="Cys3His CCG1-Required  
 Encoded on BAC clone RP5-842K24 (AL050310)  
 The human CHCR (Cys3His CCG1-Required) protein is highly  
 related to EXP/MBNL (Y13829, NM\_021038, AF401998) and MBLN  
 (NM\_005757, AF061261), which together comprise the human  
 Muscblind family"  
 /codon\_start=1  
 /product="hCHCR-G"  
 /protein\_id="BAB85648.1"  
 /db\_xref="GI:19032345"  
 /translation="MTAVNVALIRDTKMLTEVCREFORGTCSRADADCKFAHPPRVC  
 HVENGRVAVCFDSLKGRCTRENCKYLHPHPLKTLQLEINGRNLIQQKTAAMFAQQM  
 QLMQLQAQMSLGLSFPMTSPANPPMFPNFIYPHGMGLVPAELVPVLIIPGNPP  
 LAMPVAVGPKLMRSKLEVCREFORGNCTRENGDCRYAHPDASMI EASDNTVTICMD  
 YIKGRCSREKCKYFHPHQLARLKAHHQMNHSAASAMALQPGTLQIIPKRSALFKP  
 NGATPVFNFVFCQQALNQLPQPAFIPAGPIILCMAPASNIVPMHGGATPTVTSAA  
 TTPATSVFPAAPTGNQLKF"  
 misc\_feature 137..196  
 /gene="CHCR"  
 /note="Cys3His, zinc finger  
 Encoded on BAC clone RP5-842K24 (AL050310)"  
 misc\_feature 239..292  
 /gene="CHCR"  
 /note="Cys3His, zinc finger  
 Encoded on BAC clone RP5-842K24 (AL050310)"  
 misc\_feature 617..676  
 /gene="CHCR"  
 /note="Cys3His, zinc finger  
 Encoded on BAC clone RP5-842K24 (AL050310)"  
 misc\_feature 725..778  
 /gene="CHCR"  
 /note="Cys3His, zinc finger  
 Encoded on BAC clone RP5-842K24 (AL050310)"  
 3'UTR 1145..2659  
 /gene="CHCR"  
 polyA\_site 1606  
 /gene="CHCR"

3'UTR 1145..2659  
 /gene="CHCR"  
 polyA\_site 1606  
 /gene="CHCR"  
 /note="Encoded on BAC clone RP5-842K24 (AL050310);  
 PolyA\_site#1 used by CHCR EST clone PLACE1010202  
 (AK002178)"  
 polyA\_site 2660  
 /gene="CHCR"  
 /note="Encoded on BAC clone RP5-842K24 (AL050310);  
 PolyA\_site#2 used by CHCR EST clone DKFPz4346222  
 (AL133625)"  
 BASE COUNT 817 a 570 c 525 g 789 t  
 ORIGIN  
 1 aattcatttt taatccttta atagtcacca gtaattattgt cctaaagagg gtacattgga  
 61 ttttaatttt gctttcaata tgacgggtgt caatgttgcc ctgattcgtg ataccagtg  
 121 gctgacttta gaagctctga gagaatttca gagaggaact tgctctcogag ctgatgcaga  
 181 ttgcaagttt gccatccacc caagagtttg ccatgtggaa aatggtcgtg tgtggtcctg  
 241 ttttgattct ctaaagggtc ggtgtaccgg agagaactcg aagtaccttc accctcctcc  
 301 acacttaaaa acgcagctgg agattaatgg gcggaacaat ctgattcaac agaagaactgc  
 361 cgcagccatg ttgcgccagc agatgcaact tatgtcccaa aacgctcaaa tgcatacact  
 421 tggttctttt cctatgactc catcaattcc agctaactct cccatggctt tcaatcctta  
 481 cataccacat cctgggatgg gctcgtttcc tgcagaactt gtaccaataa caactgttct  
 541 gatcctgga aaaccaactc ttgcaatgcc aggagctgtt ggcacaaac tgatcgtctc  
 601 agataaactg gaggtttgcc gagaatttca cgtggaaat tgcacctcgt gggagaatga  
 661 ttgcgcctat gctcacoccta ctgatgcttc catgatgaa gcgagtgata atactgtgac  
 721 aatctgcatg gattacatca aaggtcgtat ctgcggggag aatgcaagt actttcatcc  
 781 tctctgcacc ttgcaagcca gactcaaggc agctcatcat cagatgaacc attcagctgc  
 841 ctctgcctat gccctgcagc ctggtacact gcaactgata ccaagaagat cagcactgga  
 901 aaagcccaat ggtgccacc cgtcttttaa tcccactgtt tccaactgcc aacagctctc  
 961 gactaacctg caegtccacc agcccgcatc tctccctgca gggcaaatc tgtgcctgct  
 1021 acccgcttca aatattgtgc caatgatgca cgtgctcaac cctaccactg tctctcagc  
 1081 aacaacacct gccaccagcg ttccglttgc tgcaccaact acaggaactc agctgaaatt  
 1141 ctgacaagca gaggatgga gtaacagaat ctctccatgg aaacctccat atggccttc  
 1201 tatatatatt ctgctatgct ttattctaac aacaacaaca taagctgtt gcagcgaatg  
 1261 tattaagca agcaacctg ccagccagca aatcaataa aaaataaag cattaanaat  
 1321 caatggagat gtaaaacaaa cacaaataga aaactagtaa ctaccatcca tccattttga  
 1381 atatacaagc agaacatgac cataaaattt ggtaactgtg tacattactc tttgtattt  
 1441 totaataaac atgtaagtgt tattccaca gtgagctttt gcttactat atacattott  
 1501 ggtggataaa ttgtctactt gttttgagc ttttacctta ctattttgt tacagaatag  
 1561 tctattgggt tgatcagga tgtaacaaat atattcagta coattctgt tgttattgt  
 1621 tttgtgctgt tttatgctt ttacatctg tagtgtttg ctgtatagt gttgtttgt  
 1681 atttcaacta aagtgttatt agtggggac agaagtatat gtgcttaaga acatgacagg  
 1741 ttcattgaaa tatgctctct tcttttagaa tttttctgta ggtttcttgg gactgacatt  
 1801 taaaacgctt cacttttgaa tgtgcacaaa accctgctca taactgcat gtgtataatt  
 1861 tgcactgca gatctgatgt tgcataaac aatcaaatc ctagatattt taagaagaga  
 1921 aataattacc tgcacaagc agagaacttc ataaaacatt aaccctaat tcaacttctt  
 1981 taaatgctt agcaaatag actttacctt taaatgaatt tctcagatt tactactaaa  
 2041 a gattttttg tgtgtggct tgagatccc atctccataa  
 2101 c catttgcat ctccagctga gaaactgta cttggagctg  
 2161 a gttataatc aaacggagag atggggcat ggagatggt  
 2221 t gtgtaaaccc atggcaatgt cacctttac acaaatgcc  
 2281 t catgctctt agactactct ttgaataca agtaagctgc  
 2341 aatctagcaa aagtcagca ggtgaaaga gaattggtg caaatgaga ctccctccc  
 2401 caaatggaca gtctctctg ttgatcacag agggagctg agtacagctc tggagaatg  
 2461 gctaggacag gaaacagga agcacttaca attattctt gatttatca aaagaactgc  
 2521 gaaagatgt tgtagtgtc tttagcttc gttcaactga gtttctttt gtaaacagt  
 2581 tcagtgaagc agaaagcacc tgtgatata ggcaagtgc cccctgccca aactttaaca  
 2641 tcagaccctc tcacatcata aaaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa  
 2701 a //



# Quick Example (answer)

```
[babelfish]$ perl ex1.pl
Sequence length is 2701
1st ATG is at 80
features are:
source (1..2701)
gene (1..2701)
5'UTR (1..79)
CDS (80..1144)
misc_feature (137..196)
misc_feature (239..292)
misc_feature (617..676)
misc_feature (725..778)
3'UTR (1145..2659)
polyA_site (1606..1606)
polyA_site (2660..2660)
```

# Toolkit Components

- Biological Sequences (nt, aa) & seq file parsers
- Sequence Features (locations), Annotations
- Sequence Databases (local and remote)
- Seq Database Search Results (BLAST, FASTA)
- Alignments (pairwise and multi-sequence)
- Protein Structures

# (more) Toolkit Components

- Coordinate System Mapping (chrom -> protein)
- Sequence Variation
- Maps (markers, genetic, FPC)
- Trees, TreeO (some statistics methods implemented)
- Populations (some statistics implemented) & Pedigrees

# Toolkit Components - Analysis

- Run applications locally (BLAST, EMBOSS, ClustalW, RepeatMasker)
- Connect to remote application systems (NCBI Web Blast, SOAP-Based (EBI) )
- Analysis Pipeline - BioPipe

# (new) Toolkit Components

- Graphics - rendering of sequences with features (tracks)
- Ontology handling (GO, SOFA)
- Simple Relational DB for Features and sequences (Bio::DB::GFF)
- Complex RDBMS schema for full set of sequences, features (BioSQL)

# Simple Interface

```
use Bio::Perl;
picks the filename up from the command line
my $filename = shift;

reads the file assuming it is a sequence
my $seq_object = read_sequence($filename);

we print out to the screen somethings about the
sequence

print "Sequence name is ",
 $seq_object->display_id,"\n";
print "First 5 residues are ",
 $seq_object->subseq(1,5),"\n";
```

# More complex interface

```
use Bio::SeqIO;
picks the filename up from the command line
my $filename = shift;

reads the file assuming it is a sequence
my $parser = new Bio::SeqIO(-file => $filename,
 -format=> 'fasta');

my $seq_object = $parser->next_seq;

we print out to the screen somethings about the
sequence

print "Sequence name is ",
 $seq_object->display_id,"\n";
print "First 5 residues are ",
 $seq_object->subseq(1,5),"\n";
```

# What's the point?

```
use Bio::SeqIO;
picks the filename up from the command line
my $filename = shift;

reads the file assuming it is a sequence
my $parser = new Bio::SeqIO(-file => $filename,
 -format=> 'genbank');

iterate through all the sequences in the file
while(my $seq_object = $parser->next_seq) {
 # we print out to the screen somethings about the
 # sequence
 print "Sequence name is ",
 $seq_object->display_id,"\n";
 print "First 5 residues are ",
 $seq_object->subseq(1,5),"\n";
}
```

# Getting Sequences from GenBank

- Through Web Interface Bio::DB::GenBank (don't abuse!!)
- Alternative is to download all of genbank, index with Bio::DB::Flat (will be **much** faster in long run)

# Simple Sequence Retrieval

```
use Bio::Perl;

my $seq = get_sequence('genbank', $acc);

print "I got a sequence $seq for $acc\n";
```

# Sequence Retrieval Script

```
#!/usr/bin/perl -w
use strict;

use Bio::DB::GenPept;
use Bio::DB::GenBank;
use Bio::SeqIO;

my $db = new Bio::DB::GenPept();
my $db = new Bio::DB::GenBank(); # if you want NT seqs
use STDOUT to write sequences
my $out = new Bio::SeqIO(-format => 'fasta');

my $acc = 'AB077698';
my $seq = $db->get_seq_by_acc($acc);
if($seq) {
 $out->write_seq($seq);
} else {
 print STDERR "cannot find seq for acc $acc\n";
}
$out->close();
```

# Sequence Retrieval from Local Database

```
use Bio::DB::Flat;

my $db = new Bio::DB::Flat(-directory => '/tmp/idx',
 -dbname => 'swissprot',
 -write_flag => 1,
 -format => 'fasta',
 -index => 'binarysearch');

$db->make_index('/data/protein/swissprot');
my $seq = $db->get_Seq_by_acc('BOSS_DROME');
```

# Parsing BLAST

- 3 components
  - Result (one per Query), database stats, search parameters, cutoffs, etc
  - Hit (sequences with high scoring alignments) (1..N per Query)
  - HSP (High-Scoring Segment Pair) - alignment (1..N per Hit)

# BLAST Report

Copyright (C) 1996-2000 Washington University, Saint Louis, Missouri USA.  
All Rights Reserved.

Reference: Gish, W. (1996-2000) <http://blast.wustl.edu>

Query= BOSS\_DROME Bride of sevenless protein precursor.  
(896 letters)

Database: wormpep87  
20,881 sequences; 9,238,759 total letters.  
Searching...10....20....30....40....50....60....70....80....90....100% done

| Sequences producing High-scoring Segment Pairs:             | High Score | Smallest Sum P(N) | Probability N |
|-------------------------------------------------------------|------------|-------------------|---------------|
| F35H10.10 CE24945 status:Partially_confirmed TR:Q20073...   | 182        | 4.9e-11           | 1             |
| M02H5.2 CE25951 status:Predicted TR:Q966H5 protein_id:...   | 86         | 0.15              | 1             |
| ZC506.4 CE01682 locus:mgl-1 metatrophic glutamate recept... | 91         | 0.18              | 1             |
| F23D12.2 CE05700 status:Partially_confirmed TR:Q19761 ...   | 73         | 0.45              | 3             |

>F35H10.10 CE24945 status:Partially\_confirmed TR:Q20073  
protein\_id:AAA81683.2  
Length = 1404

Score = 182 (69.1 bits), Expect = 4.9e-11, P = 4.9e-11  
Identities = 75/315 (23%), Positives = 149/315 (47%)

Query: 511 YPFLFDGESVMFWRKMDTWVATGLTAAILGLIATLAILVFIVVRISLGDVFEQNPPTS 570  
Y +F+ + WR +V L ++ + +A+LV ++V++ L V +GN + I  
Sbjct: 1006 YQSVFEHITGHRDHPHNYVLLALITVLV--VVAIAVLVLVLKLYLR-VVKGNSLGI 1062

Query: 571 LLLLSLILVFCFSVPYSIEYVGEQRNSHVTFEDAQTNTLCAVRVFMIMLVYCFVFSLLL 630  
LL+ +I++ YS + F+ +++C +RV + L Y F +++  
Sbjct: 1063 SLLIGIIL-----YSTAFF-----FVFDPT---DSVCRRLVILHGLGYTICFGVMI 1106

Query: 631 CRAVMLASIGSEG-GFLSHVNGYIQAIVCAFVVAQVGMVQLLVVMHVASETVSCENIY 689  
+A L + + G G H++ + ++ F V Q+ +S+ + +++ V N+  
Sbjct: 1107 AKATQLRNAETLGFGTAIHISFWNYWLLFFIVGVQIALSISWFLEPFMSTIGVIDTNVQ 1166

Query: 690 YGRWLWGLLAY---DF--ALLCCV GALIPSIYRS-QRNYREGILIVIGSVLIMVIWVAWI 74  
G + + +F +L + I + R+ +RNY+E ++ +VL WVAWI  
Sbjct: 1167 RMMCTMGKVEFVVSFNFYVMILIFMALFISMLNRNRIKRNKYEKTKWLLYSTVLCFFPTWVAWI 12

Query: 744 ALSLFGD-EWRDAAIPLGLQASGWAVLVGILI-PRTFIVRGIERSDIA---QALPSLTS 79  
L L D E+RD I + L A +L+G L P+ ++++ E +A P+ T  
Sbjct: 1227 TLYLVLDHEFRDVTIVVELVACA-TILLGFLFGPKIYILL-SYEPVVVAFKRDPFPNHTD 12

Query: 799 LAF AQNNQYSSEQSV 813  
L F +++ S+++V  
Sbjct: 1285 L-FEKDDDLPSQRAV 1298

Score = 39 (18.8 bits), Expect = 1.4e-09, Sum P(2) = 1.4e-09  
Identities = 9/28 (32%), Positives = 17/28 (60%)

Query: 508 NRRYPFLFDGESVMFWRKMDTWVATGL 535  
N++ P F GE + F+R + +A+G+  
Sbjct: 427 NKQDPPEFAGERLQFYRGTENILLASGM 454

>M02H5.2 CE25951 status:Predicted TR:Q966H5 protein\_id:AAK84552.1  
Length = 326

Score = 86 (35.3 bits), Expect = 0.16, P = 0.15  
Identities = 41/151 (27%), Positives = 70/151 (46%)

Query: 525 IKMDTWVATGLTAAILGLIATLAILVFIVVRISLGDVFEQNPPTSILLLSLI-LVFCSF 58  
I + W ++ L A +L T +L F + + +F+ N T +L+L ++ L F F  
Sbjct: 114 IGLGLWASSCLVAMLL---VTNRLLDF-TFKSAANFMFDKNTFLVLILPTIYGLYFAIF 16

Query: 584 VP---YSIEYVGEQRNSHV-TFEDAQTNTLCAVRVFMIMLVYCFVFSLLLCR--AVML- 63  
P YS +Y+G + + E A+ N I+ V CF++ + C A M  
Sbjct: 170 TPPLLYSSKYLGWFFDFPFIFQGETAEYNYPHLANNIIVVAVTCFLYIMFCCALGAEMKN 22

Query: 637 ASIGSEGGFLSHVNGYIQAIV-ICAFSVVAQV 666  
+ GSE + +IQ+V IC F++ A +  
Sbjct: 230 VNTGSES-YKRSKQIFIQSVLICGFNLATSL 259

# BLAST Parsing Script

```
use Bio::SearchIO;
my $cutoff = '0.001';
my $file = 'BOSS_Ce.BLASTP',
my $in = new Bio::SearchIO(-format => 'blast',
 -file => $file);
while(my $r = $in->next_result) {
 print "Query is: ", $r->query_name, " ",
 $r->query_description," ",$r->query_length," aa\n";
 print " Matrix was ", $r->get_parameter('matrix'), "\n";
 while(my $h = $r->next_hit) {
 last if $h->significance > $cutoff;
 print "Hit is ", $h->name, "\n";
 while(my $hsp = $h->next_hsp) {
 print " HSP Len is ", $hsp->length('total'), " ",
 " E-value is ", $hsp->evaluate, " Bit score ",
 $hsp->score, " \n",
 " Query loc: ",$hsp->query->start, " ",
 $hsp->query->end," ",
 " Subject loc: ",$hsp->hit->start, " ",
 $hsp->hit->end,"\n";
 }
 }
}
```

# Script Results

```
Query is: BOSS_DROME Bride of sevenless protein precursor.
896 aa
Matrix was BLOSUM62
Hit is F35H10.10
HSP Len is 315 E-value is 4.9e-11 Bit score 182
Query loc: 511 813 Subject loc: 1006 1298
HSP Len is 28 E-value is 1.4e-09 Bit score 39
Query loc: 508 535 Subject loc: 427 454
```

# FASTA Parsing Script

```
use Bio::SearchIO;
my $cutoff = '0.001';
my $file = 'BOSS_Ce.FASTP',
my $in = new Bio::SearchIO(-format => 'fasta',
 -file => $file);

while(my $r = $in->next_result) {
 print "Query is: ", $r->query_name, " ",
 $r->query_description, " ", $r->query_length, " aa\n";
 print " Matrix was ", $r->get_parameter('matrix'), "\n";
 while(my $h = $r->next_hit) {
 last if $h->significance > $cutoff;
 print "Hit is ", $h->name, "\n";
 while(my $hsp = $h->next_hsp) {
 print " HSP Len is ", $hsp->length('total'), " ",
 " E-value is ", $hsp->evaluate, " Bit score ",
 $hsp->score, " \n",
 " Query loc: ", $hsp->query->start, " ",
 $hsp->query->end, " ",
 " Subject loc: ", $hsp->hit->start, " ",
 $hsp->hit->end, "\n";
 }
 }
}
```

# Script Results

Query is: BOSS\_DROME Bride of sevenless protein precursor.  
896 aa  
Matrix was BL50  
Hit is F35H10.10  
HSP Len is 728 E-value is 6.8e-05 Bit score 197.9  
Query loc: 207 847 Subject loc: 640 1330

# SearchIO system

- Parsing DB Search Results
- BLAST (WU-BLAST, NCBI, XML, PSIBLAST\*)
- FASTA
- HMMER
- Exonerate, WABA, BLAT

# Graphics

- Simple code to render Sequence with 'tracks'
- Basic component of Generic Genome Browser (GBrowse)
- Highly customizable, extensible

# Generate a Graphics Panel

```
use Bio::Graphics;
use Bio::SeqIO;
use Bio::SeqFeature::Generic;

my $file = shift
 or die "provide a sequence file as the argument";
my $io = Bio::SeqIO->new(-file=>$file)
 or die "couldn't create Bio::SeqIO";
my $seq = $io->next_seq
 or die "couldn't find a sequence in the file";

my @features = $seq->all_SeqFeatures;

sort features by their primary tags
my %sorted_features;
for my $f (@features) {
 my $tag = $f->primary_tag;
 push @{$sorted_features{$tag}}, $f;
}
my $panel = Bio::Graphics::Panel->new(
 -length => $seq->length,
 -key_style => 'between',
 -width => 800,
 -pad_left => 10,
 -pad_right => 10);

$panel->add_track(arrow =>
 Bio::SeqFeature::Generic->new(-start => 1,
 -end => $seq->length),
 -bump => 0,
 -double=>1,
 -tick => 2);

$panel->add_track(generic =>
 Bio::SeqFeature::Generic->new(-start => 1,
 -end => $seq->length,
 -bgcolor => 'blue',
 -label => 1,));

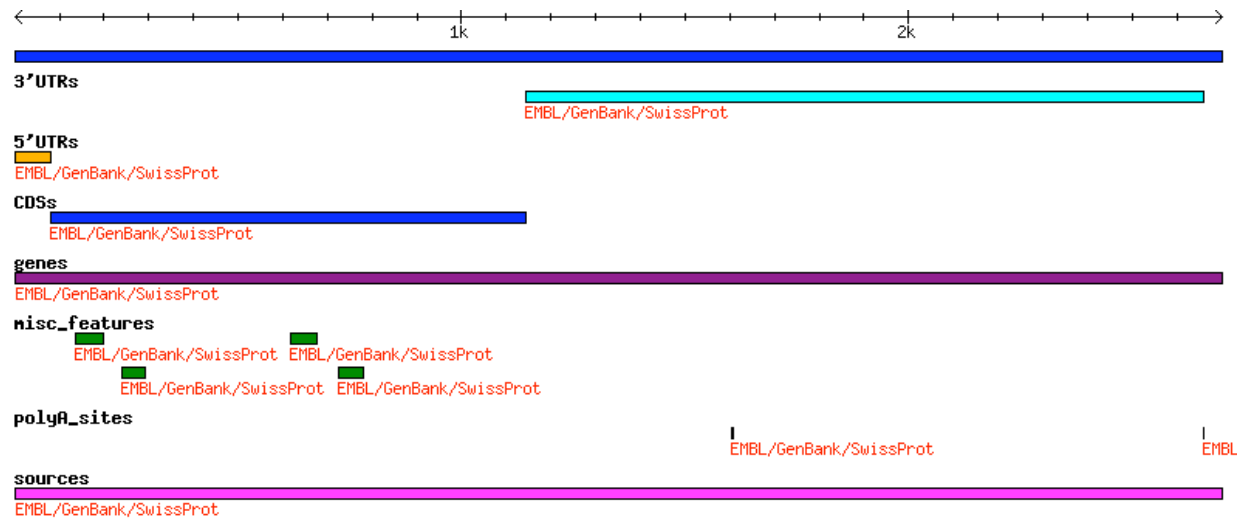
general case
my @colors = qw(cyan orange blue purple green
 chartreuse magenta yellow aqua);
my $idx = 0;

for my $tag (sort keys %sorted_features) {
 my $features = $sorted_features{$tag};
 $panel->add_track($features,
 -glyph => 'generic',
 -bgcolor => $colors[$idx++ % @colors],
 -fgcolor => 'black',
 -font2color => 'red',
 -key => "${tag}s",
 -bump => +1,
 -height => 8,
 -label => 1,
 -description => 1,
);
}

print $panel->png;
```

# Graphics output

```
[babelfish]$ perl graphics.pl AB077698.gbk > AB077698.png
```



# How do I learn how to really use it?

- Overview documentation
- Tutorials ([bptutorial.pl](#))
- Every module (should) have SYNOPSIS code
- HOWTOs
- Join mailing list
- Books in progress...

# Where to get help

- Online documentation -  
<http://doc.bioperl.org>
- On your computer: `perldoc Bio::Seq`
- Tutorial: `perldoc bptutorial.pl` or on  
<http://bioperl.org> website
- HOWTOs linked of bioperl website as well